**CS381-37: Project 1.1 (JAVA)**

**Yida Tao**

**Due date: Feb. 07, 2018**

Algorithm Steps:

step 0: open input and output files

step 1: numRows, numCols, minVal, maxVal <-- get from input file

dynamically allocate the hist array and initialize to 0

step 2: // process the input file from left to right and top to bottom

p(i,j) <- read from input // you must read one integer at a time

hitogram[p(i,j)]++

step 3: repeat step 2 until the file is empty

step 4: output histogram array to output file // follow the format given

step 5: close input file and output file

**Source Code**

**import** java.io.\*;

**import** java.util.Scanner;

**public** **class** Project1 {

**public** **static** **void** main(String[] args) {

Scanner sc = **null**;

FileOutputStream fos = **null**;

String s;

**int** numRows = 0;

**int** numCols = 0;

**int** minVal = 0;

**int** maxVal = 0;

**int** hist[] = **null**;

//step 0: open input and output files

**try** {

sc = **new** Scanner(**new** File(args[0]));

} **catch**(ArrayIndexOutOfBoundsException e){

e.printStackTrace();

System.***out***.println("No argument");

} **catch** (FileNotFoundException e) {

e.printStackTrace();

System.***out***.println("Cant find the file" + args[0]);

}

//set output to output file

**try** {

fos = **new** FileOutputStream(args[1]);

System.*setOut*(**new** PrintStream(fos));

} **catch** (FileNotFoundException e) {

e.printStackTrace();

}

//step 1: initialize

**try** {

numRows = sc.nextInt();

numCols = sc.nextInt();

minVal = sc.nextInt();

maxVal = sc.nextInt();

hist = **new** **int**[maxVal+1];

//step 2&3: read from file, until file empty

**while**(sc.hasNext()) {

hist[sc.nextInt()]++;

}

} **catch** (Exception e) {

e.printStackTrace();

} **finally** {

sc.close();

}

//step 4: output histogram array to output file

**for**(**int** i = 0; i < maxVal; i++){

System.***out***.println(i + " " + hist[i]);

}

//step 5: close input file and output file

**try** {

fos.close();

System.*setOut*(**new** PrintStream(**new** FileOutputStream(FileDescriptor.***out***)));

} **catch** (IOException e) {

e.printStackTrace();

}

System.***out***.println("Done");

}

}

**Input**

31 40 0 9

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1

2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 7 2 3 2 3 5 3 2 3 1 2 1 2 1 2 1 2 1 0

1 2 1 2 1 2 2 3 2 3 2 3 2 1 2 1 2 4 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1

0 3 1 2 1 2 1 2 1 2 1 2 1 2 4 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 0 1

1 2 1 2 1 2 4 3 2 3 2 3 2 3 5 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 1 2 1 2 1 2 1 2 1 0

0 3 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 9 5 0 1

1 2 4 2 1 2 4 3 2 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 2 1 2 9 2 1 0

0 1 2 1 2 3 5 4 5 7 7 8 7 7 8 7 7 7 7 8 7 8 7 7 7 8 7 7 7 7 7 7 5 2 9 3 2 1 2 1

0 2 3 1 2 1 2 1 5 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 7 6 1 2 1 2 1 2 1

1 2 3 3 2 1 2 1 6 9 9 9 7 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 8 7 5 1 2 1 4 1 1 0

0 0 1 1 2 2 1 2 3 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 8 7 5 1 2 1 2 5 1 0

1 2 3 3 2 1 2 1 5 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 0 9 9 9 9 8 7 2 1 2 1 2 1 1 0

0 9 9 1 2 1 2 1 5 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 0 9 0 9 9 9 8 7 1 2 1 5 1 1 0 0

0 1 2 3 3 1 2 1 5 8 9 9 9 9 0 9 1 9 0 9 9 9 9 9 9 0 0 9 9 9 9 8 2 1 2 1 2 4 5 0

0 0 1 1 2 2 1 2 6 9 9 9 9 0 0 9 9 0 0 9 9 9 9 9 9 9 9 9 9 9 8 7 1 1 2 1 2 1 0 0

0 0 1 1 2 2 4 5 6 9 9 9 9 9 9 9 9 1 1 9 9 9 9 9 9 9 9 9 9 9 8 7 1 2 1 2 1 0 0 0

1 2 3 3 4 4 5 5 7 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 2 9 9 9 9 8 7 2 1 2 1 2 1 5 0

0 0 1 1 2 2 4 5 6 9 9 9 9 9 9 9 9 9 9 9 9 9 9 0 9 9 9 9 9 9 8 7 1 1 2 1 2 1 0 0

0 1 2 3 3 3 3 5 5 9 9 7 9 9 9 9 9 9 9 9 9 9 9 0 0 9 9 9 9 9 9 1 1 2 1 9 1 4 0 0

0 2 3 1 1 3 3 4 5 8 8 8 8 8 8 8 8 8 9 8 8 8 8 7 8 8 8 8 8 8 8 7 1 1 2 1 2 9 0 1

0 1 2 1 2 3 5 6 5 9 9 7 9 7 7 8 7 8 8 7 9 7 9 7 7 7 2 9 7 7 7 7 0 0 1 1 2 1 2 1

0 3 1 2 1 2 4 3 2 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 2 1 2 1 2 0 1

1 2 1 2 1 2 4 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 1 2 1 2 4 8 1 2 1 0

0 3 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 0 1

1 2 1 2 1 2 4 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 1 2 1 2 3 2 1 2 1 0

0 3 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 0 1

0 1 2 1 2 1 2 1 2 1 2 1 2 1 5 1 2 1 2 1 2 1 2 1 5 1 2 1 2 1 2 1 2 1 2 5 2 1 2 1

2 3 4 3 2 3 2 3 5 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 1 2 1 2 1 2 1 2 1 0

1 2 1 2 5 2 2 3 2 3 2 3 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

**Output**

0 146

1 271

2 309

3 99

4 19

5 54

6 30

7 46

8 61

9 205